Aufgabe 1: Superstar

Team-ID: 00920

Team-Name: Paddy Jo

Bearbeiter/-innen dieser Aufgabe:
Patrick Müller, Josua Kugler

26. November 2018

Inhaltsverzeichnis

1 Lösungsidee **1**

2 Umsetzung **2**

3 Beispiele **2**
3.1 Beispiel 2
3.2 Beispiel 3
3.3 Beispiel 3
3.4 Beispiel 3
3.5 Beispiel 3

4 Quellcode **3**

1 Lösungsidee

Wird eine Anfrage gestellt, so erhält man die Information, dass eine Person, nennen wir sie A, einer anderen Person B folgt oder nicht folgt.

Wenn A B folgt, dann ist A mit Sicherheit kein Superstar. Wenn A B nicht folgt, dann ist B kein Superstar. Daher kann man bei jeder Anfrage eine Person als Superstar ausschließen. Die andere Person ist nun unser Superstarkandidat. In der nächsten Anfrage wird nun gefragt, ob der Superstarkandidat einer anderen noch nicht ausgeschlossenen Person folgt, etc. Dabei werden alle Anfragen gespeichert. Schließlich bleibt nur noch eine Person als Kandidat übrig. Die Ermittlung eines solchen Kandidaten benötigt also $n - 1$ Anfragen, wenn n die Anzahl der Teilnehmer ist. Für den Kandidaten muss nun noch verifiziert werden, dass er tatsächlich ein Superstar sein kann. Das ist genau dann der Fall, wenn alle anderen Personen dem Kandidaten folgen und er keiner anderen Person folgt. Daher wird zuerst in den gespeicherten Anfragen nach Anfragen gesucht, bei denen eine Person dem Kandidaten folgt. Danach werden weitere Anfragen gestellt, bis entweder bestätigt wurde, dass alle Personen dem Kandidaten folgen oder eine Person ihm nicht folgt und es also keinen Superstar gibt. Dieser Teil benötigt erneut maximal $n - 1$ Anfragen.

Wenn bis hierher nicht gezeigt wird, dass der Superstarkandidat nicht Superstar ist, wird nun in den gespeicherten Anfragen nach Personen gesucht, denen der Kandidat folgt. Danach wird solange mit Anfragen überprüft, ob es jemanden gibt, dem der Kandidat folgt bis entweder bestätigt wurde, dass er niemandem folgt und er tatsächlich Superstar ist oder es eine Person gibt, der er folgt und er dementsprechend kein Superstar ist. Auch hierfür werden maximal $n - 1$ Anfragen benötigt. Da während der Ermittlung des Kandidaten bereits mindestens eine Aussage über den Kandidaten getroffen wird. Diese Anfrage muss also während der Verifizierung des Kandidaten nicht mehr gestellt werden. Insgesamt lässt sich also mit maximal $3n - 4$ Anfragen ermitteln, ob es einen Superstar gibt und wenn ja, wie er heißt.

2 Umsetzung

Zunächst wird die als Kommandozeilenargument übergebene Textdatei eingelesen und es werden zwei Listen erstellt. Eine Liste (`grouplist`) enthält alle Teilnehmer der Gruppe. Diese Liste wird von allen Funktionen benutzt. Die andere Liste (`followlist`) enthält alle Beziehungen zwischen Personen. Wenn eine Person A einer Person B und eine Person C einer Person D folgt, wird dies in der Form `[[A,B],[C,D]]` dargestellt. Auf diese Liste darf nur die Funktion `Anfrage` zugreifen.

Die Funktion `getsuperstarcandidate` schließt mittels einer Anfrage stets eine Person aus, die nicht Superstar ist und fügt diese zur Liste `nsuperstar` hinzu, die andere Person wird `superstarcandidate`. Wird eine Anfrage der Form `Anfrage(A,B)` gestellt und mit False beantwortet, so wird die Liste `[A,B]` unter `Anfragenliste[0]` gespeichert, andernfalls unter `Anfragenliste[1]`. Dann ruft sich die Funktion `getsuperstarcandidate` selbst mit der Person `superstarcandidate` und einer weiteren Person auf. Die zweite Person darf nicht in der Liste `nsuperstar` sein, damit man nicht Personen ausschließt, die bereits ausgeschlossen sind. Auch von diesen beiden Personen wird wieder eine ausgeschlossen und zur Liste `nsuperstar` hinzugefügt. Sobald alle Personen aus `grouplist` bis auf eine ausgeschlossen sind, die Länge von `nsuperstar` also um eins geringer als die Länge von `grouplist` ist, muss noch überprüft werden, ob die übrige Person tatsächlich Superstar sein kann. Dies geschieht mit der Funktion `verifysuperstar`.

Diese überprüft zunächst, welche gespeicherte Anfragen es gibt, bei denen eine Person dem Kandidaten folgt, indem mit einem for-loop über alle positiv beantworteten Anfragen, also `Anfragenliste[1]`, iteriert wird. Jede Person, die dem Kandidaten folgt, wird zur Liste `candidatefollowers` hinzugefügt. In den gespeicherten Anfragen gibt es keine Anfragen, die den Kandidaten ausschließen, da er ansonsten bereits in der Liste `nsuperstar` enthalten wäre. Diese Anfragen werden nur untersucht, um die Zahl der noch zu stellenden Anfragen zu minimieren. Danach werden weitere Anfragen gestellt, indem mithilfe eines for-loops über jede Person aus der `grouplist` iteriert wird, die sich nicht bereits in `candidatefollowers` befindet. Sollte es eine Person geben, die dem Kandidaten nicht folgt, so gibt es keinen Superstar in dieser Gruppe, da alle anderen Personen bereits davor ausgeschlossen wurden. In diesem Fall gibt die Funktion mittels eines return-statements diese Information zurück. Wenn dies nicht der Fall ist, wird nun in den gespeicherten Anfragen nach Personen gesucht, denen der Kandidat nicht folgt, indem über alle negativ beantworteten Anfragen iteriert wird. Personen, denen der Kandidat nicht folgt, werden zur Liste `candidatenotfollows` hinzugefügt. Danach wird solange mit Anfragen der Form `Anfrage(candidate,person)` mit den Personen, die nicht in `candidatenotfollows` enthalten sind, überprüft, ob es jemanden gibt, dem der Kandidat folgt, bis entweder bestätigt wurde, dass er niemandem folgt, dann ist er tatsächlich Superstar und die Funktion gibt dies zurück, oder jemand gefunden wird, dem er folgt. Dann ist er nicht Superstar der Gruppe und es gibt keinen Superstar in dieser Gruppe. Da die Funktion `getsuperstarcandidate` das Ergebnis der `verifysuperstar`-Funktion zurückgibt, erhält man somit das gewünschte Ergebnis als Rückgabewert von `getsuperstar`. Die Kosten werden berechnet, indem bei der Funktion `Anfrage` bei jedem Aufruf die Variable `Anfn` um 1 erhöht wird.

3 Beispiele

3.1 Beispiel

Selena folgt Justin, also ist Selena kein Superstar
Justin folgt Hailey nicht, also ist Hailey kein Superstar
Hailey folgt Justin
Justin folgt Selena nicht
Justin ist der Superstar dieser Gruppe!
Kosten: 4€
Kommentar: Zunächst wurde mithilfe der Funktion `getsuperstarcandidate` ermittelt, das sowohl Selena als auch Hailey nicht Superstar sein können. Dann wird noch überprüft, ob alle Justin folgen: Von Selena weiß man bereits, dass sie Justin folgt (1. Anfrage), sodass man nur noch überprüfen muss, ob Hailey Justin folgt. Schließlich muss noch gezeigt werden, dass Justin keiner anderen Person folgt. Man weiß bereits, dass Justin Hailey nicht folgt, daher muss nur noch Selena überprüft werden. Es stellt sich heraus, dass Justin der Superstar dieser Gruppe ist.

3.2 Beispiel

Turing folgt Hoare, also ist Turing kein Superstar
Hoare folgt Dijkstra, also ist Hoare kein Superstar
Dijkstra folgt Knuth nicht, also ist Knuth kein Superstar
Dijkstra folgt Codd nicht, also ist Codd kein Superstar
Turing folgt Dijkstra
Knuth folgt Dijkstra
Codd folgt Dijkstra
Dijkstra folgt Turing nicht
Dijkstra folgt Hoare nicht
Dijkstra ist der Superstar dieser Gruppe!
Kosten: 9€
Kommentar: Dieses Beispiel folgt dem gleichen Schema wie Beispiel 3.1.

3.3 Beispiel

Edsger folgt Jitse, also ist Edsger kein Superstar
Jitse folgt Jorrit nicht, also ist Jorrit kein Superstar
Jitse folgt Pia nicht, also ist Pia kein Superstar
Jitse folgt Rineke nicht, also ist Rineke kein Superstar
Jitse folgt Rinus nicht, also ist Rinus kein Superstar
Jitse folgt Sjoukje nicht, also ist Sjoukje kein Superstar
Jorrit folgt Jitse nicht
Es gibt also keinen Superstar!
Kosten: 8€
Kommentar: Man erkennt, dass zunächst angefragt wurde, ob Edsger Jitse folgt. Diese Anfrage wurde mit True beantwortet, sodass man daraus folgern kann, dass Edsger kein Superstar sein kann. Jitse ist also nun `possiblesuperstar`. Es werden nun weitere Anfragen gestellt, bis alle Personen außer Jitse ausgeschlossen sind. Es wird also die Funktion `verifysuperstar("Jitse")` aufgerufen. Diese überprüft nun mithilfe von Anfragen, ob alle Personen Jitse folgen. Dabei findet sie heraus, dass Jorrit Jitse nicht folgt. Also gibt es keinen Superstar.

3.4 Beispiel

Das Ergebnis des 4. Beispiels ist, dass Folke Superstar der Gruppe ist. Die Kosten belaufen sich auf 182€

3.5 Beispiel

Alfred folgt Bernhard, also ist Alfred kein Superstar
Bernhard folgt Carl, also ist Bernhard kein Superstar
Carl folgt David, also ist Carl kein Superstar
Alfred folgt David
Bernhard folgt David
David folgt Alfred nicht
David folgt Bernhard nicht
David folgt Carl nicht
David ist der Superstar dieser Gruppe!
Kosten: 8€
In diesem Beispiel ist der worst-case erreicht: Es gibt vier Personen, die maximale Zahl an Anfragen beträgt also $3n-4 = 3*4-4 = 8$. Das liegt daran, dass bei den Anfragen zur Ermittlung des Kandidaten nur eine einzige Anfrage gemacht wird, bei der Superstar David involviert ist.

4 Quellcode

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Paddy Jo
```

```
@title BwInf 2018 A1
"""
#Für die Übergabe von Kommandozeilenargumenten
import sys
if len(sys.argv)!=2:
    print("Benutzung: Aufgabe1.py <pfad\\zur\\datei.txt>")
    sys.exit(0)

# Einlesen der textdatei
#In der Liste "grouplist" sind alle Mitglieder aufgelistet
#In der Liste "followlist" sind alle Beziehungen aufgelistet,
#sodass die  Funktion "Anfrage" daraus einfach ermitteln kann,
#ob eine Person einer anderen folgt.
#Die Funktion "Anfrage" ist die einzige, die auf "followlist" zugreifen darf

with open(sys.argv[1]) as f:
    llist=f.readlines()
    followList=[]
    for i,line in enumerate(llist):
        if i==0:

            grouplist=line.split()
        elif i==len(llist)-1:
            pass
        else:
            followList.append(line.split())

#Anfn zählt die Anzahl der Anfragen
Anfn=0
def Anfrage(person1,person2):
    """
    Output: True, wenn person1 person2 folgt,
    ansonsten: False
    """
    global Anfn
    Anfn+=1
    for i in followList:
        if i[1] == person2:
            if i[0] == person1:
                return True
    return False

#In der Anfragenliste werden alle gestellten Anfragen mit Resultat gespeichert.
#Eine Anfrage "Anfrage(a,b)" wird als Liste [a,b] abgespeichert
#Mit False beantwortete Anfragen werden unter Anfragenliste[0] gespeichert,
#mit True beantwortete Anfragen unter Anfragenliste[1].
Anfragenliste=[[],[]]
#In der Liste nsuperstar werden alle Personen gespeichert, die nicht Superstar sein können
nsuperstar=[]


def getsuperstarcandidate(person1=None,person2=None):
    """
    - schließt mittels einer Anfrage stets eine Person aus, die nicht Superstar ist
    - ruft sich selbst mit der person superstarcandidate und einer anderen person,
      die nicht aus nsuperstar kommt und daher auch ein Superstar sein könnte, auf
    - Sobald alle Personen aus grouplist bis auf eine ausgeschlossen sind,
      wird verifysuperstar mit dieser Person (superstarcandidate) aufgerufen.
    - gibt das Ergebnis von verifysuperstar bezüglich superstarcandidate zurück
    """
    if person1 == None and person2 == None:
        person1=grouplist[0]
        person2=grouplist[1]

    if Anfrage(person1,person2):
        Anfragenliste[1].append([person1,person2])
        nsuperstar.append(person1)
        print("{} folgt {}, also ist {} kein Superstar".format(person1,person2,person1))
        superstarcandidate = person2

    else:
        Anfragenliste[0].append([person1,person2])
        nsuperstar.append(person2)
```

```
        print("{} folgt {} nicht, also ist {} kein Superstar".format(person1,person2,person2))
        superstarcandidate = person1
    #Falls nur noch ein Kandidat übrig ist:
    if len(nsuperstar)==len(grouplist)-1:
        return verifysuperstar(superstarcandidate)
    else:
        for iterateperson in grouplist:
            if iterateperson not in nsuperstar and iterateperson!=superstarcandidate:
                return getsuperstarcandidate(superstarcandidate, iterateperson)
                break


def verifysuperstar(candidate):
    """
    Output:
        - "<candidate> ist der Superstar der Gruppe!",
          wenn alle dem candidate folgen und der candidate niemandem folgt,
        - ansonsten: "Es gibt keinen Superstar!"
    """

    #test, ob alle dem candidate folgen:
    candidatefollowers = []
    for i in Anfragenliste[1]:
        if i[1] == candidate:
            candidatefollowers.append(i[0])

    for i in grouplist:
        if i != candidate and i not in candidatefollowers:
            if Anfrage(i,candidate):
                print("{} folgt {}".format(i,candidate))
                candidatefollowers.append(i)
            else:
                print("{} folgt {} nicht".format(i, candidate))
                return "Es gibt also keinen Superstar!"
    #test, ob der candidate niemandem folgt:
    candidatenotfollows = []
    for i in Anfragenliste[0]:
        if i[0] == candidate:
            candidatenotfollows.append(i[1])
    for i in grouplist:
        if i != candidate and i not in candidatenotfollows:
            if Anfrage(candidate,i):
                print("{} folgt {}".format(candidate,i))
                return "Es gibt also keinen Superstar!"
            else:
                print("{} folgt {} nicht".format(candidate,i))

    return "{} ist der Superstar dieser Gruppe!".format(candidate)

print(getsuperstarcandidate())
print("Kosten: {} Euro".format(Anfn))
print(len(grouplist)*3-4)
```

Aufgabe1.py